Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_9\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Implement bucket sort using linked list. |
| 2 | Create static tree and perform inorder, preorder and post order traversal. Also search a required node in the tree. |
|  |  |
|  |  |
|  |  |

Submitted On:

Date: 6 JAN 2022

**Task No. 1:** Implement bucket sort using linked list.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_11\_Bucket\_Sort

{

class Program

{

// Main Method

static void Main(string[] args)

{

Console.Write("Please enter the length of list : ");

int n = Convert.ToInt32(Console.ReadLine());

int[] array = new int[n];

for (int i = 0; i < n; i++)

{

Console.Write("element [{0}] = ", i);

array[i] = Convert.ToInt32(Console.ReadLine());

}

// Printing of unsorted list

Console.Write("\nUnsorted List is : { ");

for (int i = 0; i < n; i++)

{

Console.Write( array[i] + " ");

}

Console.WriteLine("}");

Console.WriteLine("\nList after applying Bucket Sort algorithm ");

Console.Write("{ ");

BucketSort(ref array);

array.ToList();

for (int i = 0; i < n; i++)

{

Console.Write(array[i] + " ");

}

Console.WriteLine("}");

Console.WriteLine("\nHave a nice Day !!!");

}

// Bucket Sort Method

public static void BucketSort(ref int[] array)

{

int minimum = int.MaxValue;

int maximum = 0;

for (int i = 0; i < array.Length; i++)

{

if (array[i] < minimum)

minimum = array[i];

if (array[i] > maximum)

maximum = array[i];

}

List<int>[] bucket = new List<int>[maximum - minimum + 1];

for (int i = 0; i < bucket.Length; i++)

{

bucket[i] = new List<int>();

}

for (int i = 0; i < array.Length; i++)

{

bucket[array[i] - minimum].Add(array[i]);

}

int k = 0;

for (int i = 0; i < bucket.Length; i++)

{

if (bucket[i].Count > 0)

{

for (int j = 0; j < bucket[j].Count; j++)

{

array[k] = bucket[i][j];

k++;

}

}

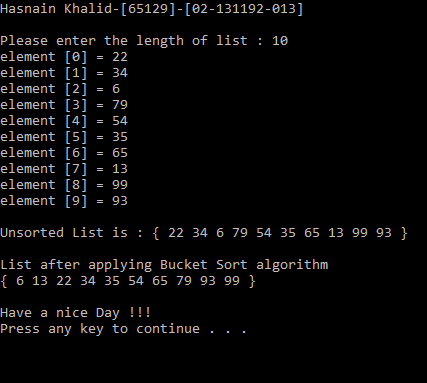
}

}

}

}

**Output:**



**Task No. 2:** Create static tree and perform inorder, preorder and post order traversal. Also search a required node in the tree.

**Solution:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApplication91

{

class BTree

{

//Node class

public class Node

{

public int data;

public Node left;

public Node right;

}

//root node of the tree

public Node root;

private int count = 0;

private void insert\_rec(int data, Node node)

{

if (node.data > data)

{

if (node.left == null)

{

node.left = new Node();

node.left.data = data;

count++;

}

else

insert\_rec(data, node.left);

}

else if (node.data < data)

{

if (node.right == null)

{

node.right = new Node();

node.right.data = data;

count++;

}

else

insert\_rec(data, node.right);

}

}

public void insert(int data)

{

if (root == null)

{

root = new Node();

root.data = data;

count++;

return;

}

insert\_rec(data, root);

}

private void preOrder\_rec(Node node)

{

if (node == null)

return;

Console.Write(node.data + " ");

preOrder\_rec(node.left);

preOrder\_rec(node.right);

}

public void preOrder()

{

Console.WriteLine("The Pre Order of binary tree is : ");

Console.Write("NLR: {");

preOrder\_rec(root);

Console.WriteLine("}");

}

private void inOrder\_rec(Node node)

{

if (node == null)

return;

inOrder\_rec(node.left);

Console.Write(node.data + " ");

inOrder\_rec(node.right);

}

public void inOrder()

{

Console.WriteLine("\nThe In Order of binary tree is : ");

Console.Write("LNR: {");

inOrder\_rec(root);

Console.WriteLine("}");

}

private void postOrder\_rec(Node node)

{

if (node == null)

return;

postOrder\_rec(node.left);

postOrder\_rec(node.right);

Console.Write(node.data + " ");

}

public void postOrder()

{

Console.WriteLine("\nThe Post Order of binary tree is :");

Console.Write("LRN: {");

postOrder\_rec(root);

Console.WriteLine("}");

}

// Count Method

public int Totalcount()

{

Console.Write("\nPress any key to display the count of all nodes : ");

char inp = Convert.ToChar(Console.ReadLine());

return count;

}

public bool search\_Node(Node node, int key)

{

if (node == null)

return false;

if (node.data == key)

return true;

bool r1 = search\_Node(node.left, key);

if (r1) return true;

bool r2 = search\_Node(node.right, key);

return r2;

}

}

class Program

{

// Main Method

static void Main(string[] args)

{

BTree btree = new BTree();

//19, 4, 31, 9, 73, 50, 15, 18, 44, 22, 10, 11, 23

btree.insert(19);

btree.insert(4);

btree.insert(31);

btree.insert(9);

btree.insert(73);

btree.insert(50);

btree.insert(15);

btree.insert(18);

btree.insert(44);

btree.insert(22);

btree.insert(10);

btree.insert(11);

btree.insert(23);

btree.preOrder();

btree.postOrder();

btree.inOrder();

Console.Write("\nPlease enter any value to be searched from your tree : ");

int node = Convert.ToInt32(Console.ReadLine());

if (btree.search\_Node(btree.root, node))

{

Console.WriteLine("Node is present in the tree");

}

else {

Console.WriteLine("Node is not present");

}

Console.WriteLine("Total count is: " + btree.Totalcount());

Console.WriteLine("\nHave a nice Day !!!");

}

}

}

**Output:**

